CSS

Rev. 5.3 del 20/10/2024

CSS

I Fogli di Stile - Cascading Style Sheets

HTML inizialmente affidava la formattazione grafica della pagina ad i vari attributi. Da HTML 3.0 in avanti, anziché incrementare il numero di attributi dei vari TAG, si è preferito seguire un'altra strada, deprecando molti attributi e lasciando all'HTML soltanto la struttura base della pagina e demandando tutta la formattazione grafica ai **Fogli di stile** (**CSS** Cascading Styles Sheets).

Le cosiddette **Proprietà di STILE** sono introdotte da un nuovo attributo **STYLE**, che consente di definire delle **proprietà ortogonali rispetto a tutti i TAG HTML**, cioè che possono essere applicati indistintamente a quasi tutti i TAG HTML, consentendo in questo modo una **netta separazione** fra i **contenuti** (scritti nel file HTML) e la **formattazione grafica** (impostata nel file .CSS)

Le **Proprietà di STILE** possono essere scritte in tre posizioni differenti:

1. Direttamente all'interno dei vari tag HTML
2. In modo compatto nella sezione di HEAD del file HTML
3. in un file esterno con estensione **.CSS**

Uno stesso tag può essere ridefinito sia su un file .CSS esterno, sia nell'intestazione della pagina, sia direttamente nel TAG. **Cascading** significa che le proprietà vengono applicate in cascata cioè **le definizione successive, più vicine al tag, nascondono le definizioni precedenti**. Cioè :

- gli stili inline sovrascrivono sia quelli della head sia quelli del file esterno.
- Le definizioni scritte nella head o in un file esterno sono equivalenti. **Prevalgono quelle scritte dopo**. Cioè se il file esterno è richiamato DOPO gli stili della HEAD, le proprietà del file esterno nascondono quelle della head e viceversa.

(1) stili INLINE definiti all'interno del tag

In questo caso le proprietà di stile sono introdotte dall'attributo **STYLE** e sono inserite all'interno di una UNICA stringa nel formato **NOME: VALORE;** col punto e virgola come separatore. Il punto e virgola dopo l'ultima voce è facoltativo.

```
<p  style = "color:red; font-size:30pt;" >  Salve a Tutti  </p>
```

(2) stili definiti nella sezione di head

```
<head>
    <STYLE>
      P { color: blue;   font-family: verdana, "times new roman" }
        /* "times new roman" ha le virgolette perchè contiene degli spazi */
    </STYLE>
</head>

<body>
     <p>  Questo viene scritto in verdana di colore blu </p>
     <p  style = "color:red; font-size:30pt;" >  Questo invece è rosso  </p>
</body>
```

(3) stili definiti in un file esterno .css

Il file esterno consente una maggiore suddivisione fra la struttura della pagina e la sua formattazione grafica.
Inoltre un file esterno potrà essere utilizzato in più pagine HTML di un stesso sito.
Le varie pagine HTML, nella sezione HEAD, dovranno richiamare il file CSS mediante il tag **LINK** :

```
<head>
       <link    rel="stylesheet"     href="mioStile.css"    >
</head>
```

I tre tipi di selettore

Le proprietà di stile possono essere definite mediante tre diversi SELETTORI :

- **SELETTORE di TAG**, cioè associati a tutti i tag di quel tipo. Ad esempio: `p {color:red}`
- **SELETTORE di CLASSE** cioè associati a tutti gli elementi che implementano una certa classe
- **SELETTORE di ELEMENTO** (o selettori assoluti) cioè associati ad un **UNICO** TAG HTML identificato tramite un apposito ID

I Selettori di classe

Consentono di creare più "**classi**" associabili a più istanze dello stesso tag oppure anche di tag differenti.
I selettori di classe sono introdotti da un ***puntino*** e potranno essere associati alle varie istanze dei tag mediante l'utilizzo dell'attributo **CLASS**

```
.mioStile1 { font-size:14pt;  color:red; }
.mioStile2 { font-size:20pt;  color:blue;  font-weight:bold;}
```

```
 <p class="mioStile1">   Questo è il mio stile 1 </p>
<div class="mioStile2">  Questo è il mio stile 2 </div>
```

E' anche possibile definire dei selettori di classe associabili *soltanto* ad un certo tipo di tag, che va espressamente indicato davanti al puntino. Esempio:

```
p.titolo { color:red;  font-size:20pt}
p.sottotitolo { color:blue  font-size :12pt}
```

```
<p class="titolo">  Questo è un titolo  </p>
<p class="sottotitolo">  Questo è un sottotitolo  </p>
```

I Selettori di Elemento (selettori assoluti)

Dato un tag avente un identificativo **ID** univoco, il Selettore di Elemento consente di associare uno stile ad un **singolo** elemento della pagina. Il selettore di Elemento è rappresentato con un simbolo ***# (pound)*** anteposto al nome del selettore.

definisce lo stile associato all'elemento HTML avente ID="BOX"

```
#BOX { color:red;  font-size:20pt }
```

```
<div id="BOX">
     <a href="Pagina2.html"> Questo link è scritto in rosso 20pt </a>
</div>
```

Annidamento dei selettori

Nella scrittura di un selettore CSS, lo spazio ha il significato di "**all'interno di**"

`#div1 a { color:red }` Tutti i tag <a> interni al tag avente id="div1" avranno colore rosso
`.class1 a { color:green }` Tutti i tag <a> interni ai tag aventi class="class1" avranno colore verde

Pseudoclassi

All'interno del selettore di classe, oltre agli stili del tag, è possibile utilizzare anche alcuni **pseudoclassi CSS** accodati al selettore principale tramite i DUE PUNTI senza spazi

```
.class1:hover{ color : red; }
```

```
<div class="class1"> Quando il mouse passa su questa scritta, il testo diventa rosso</div>
```

Analisi delle Property CSS 2.0

I campi COLOR possono essere espressi mediante i seguenti formati:

16 colori della palette VGA: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white yellow
RGB (r, g, b) Le componenti RGB sono espresse mediante numeri interi **decimali** compresi tra 0 e 255.
#RRGGBB Colore espresso mediante le componenti **esadecimali** R G B

RGBa (r, g, b, a) Solo CSS3. alfa indica la trasparenza ed è un numero decimale compreso tra **0=trasparente** e **1=solido**.
#RRGGBBAA Colore espresso mediante le componenti esadecimali R G B + canale alfa espresso in esadecimale (**FF=solido**)

Le proprietà relative ai Font

Font-Family	[[<family-name> | <generic-family>],] Nomi dei font separati da virgola. Se il browser dispone del primo font, utilizza quello, altrimenti passa al secondo e così via. Si mettono davanti i Font più specifici ed in coda quelli più generici. Non c'è valore di default, che dipende dalle impostazioni del Browser. Esempi tipici windows e macOS: Font-Family: **Arial, Helvetica, sans-serif**; con la **virgola come separatore** Font-Family: **"Times New Roman", Times, "Courier New", serif;** Notare le virgolette sui nomi con spazio.
Font-Size	font-size: <absolute-size> | <relative-size> | <length> | <percentage> dimensione del font in *punti*, in pixel, pollici, centimetri, punti web, etc <**absolute-size**> sono i 7 valori HTML: xx-small | x-small | small | medium | large | x-large | xx-large <**relative-size**> può essere larger | smaller. **<length>** valore del Font Size in pt o px <**percentage**> percentuale rispetto al valore di font-size ereditato dal genitore .
Font-weight	<normal | **bold** | 100 | 200 | 300 | **400** | 500 | 600 | **700** | 800 | 900 > bold = **700** normal=**400** I valori numerici sono utilizzabili SOLO SE disponibili nel font utilizzato (ad esempio sono disponibili in Arial ma NON in Verdana)
Font-style	<normal | *italic* | *oblique*> (oblique era più inclinato dell'italic. Oggi sembrano uguali).
Font-variant	<normal | SMALL-CAPS > SMALL CAPS SIGNIFICA CHE ANCHE LE MINUSCOLE SONO SCRITTE IN MAIUSCOLO, MA SONO LEGGERMENTE PIU' PICCOLE,
Font	Consente di specificare tutte le proprietà relative al Font in un'unica dichiarazione Es `font: 30pt bold italic` Oggi non sembra più essere supportata

Le proprietà relative ai Testi

Color	< color > Colore del testo.
Text-Align	< left | right | center | justify >. **Applicabile soltanto agli elementi di tipo BLOCK** Definisce l'allineamento orizzontale del testo interno
Vertical-Align	A differenza del precedente **NON** definisce l'allineamento verticale del testo interno ad un BlockTag ma, ***data una sequenza di tag inline / inline.block su una stessa riga***, **definisce l'allineamento reciproco di questi tag rispetto alla riga**. Per avere un perfetto allineamento su tutta la riga si può assegnare lo stesso valore (ed middle) a tutti i tag. Se si inserisce una immagine (che è inline) all'interno di una riga testuale, se si imposta il suo `vertical-align:bottom` (default), il testo circostante sarà allineato in basso (e quindi l'immagine si espanderà verso l'alto). Possibili Valori: • **baseline** (align baselines of element and parent) • **middle** (align vertical midpoint of element with baseline) • **top** (align top of element with tallest element on the line) • **bottom** (align bottom of element with lowest element on the line) • **sub** (subscript) **super** (superscript) • **text-top** (align tops of element and parent's font) relative positioning • **text-bottom** (align bottoms of element and parent's font) relative positioning

Line-Height	<normal | <number> | <length> | <percentage> > Definisce l'altezza di riga, **Normal** significa **Line-Height = Font-Size**, cioè la riga avrà una altezza pari a Font-Size. Eventuali bordi vengono tracciati sul perimetro del testo. Impostando un valore inferiore rispetto al Font-size, eventuali bordi avrebbero interferenza con il testo **percentage** è riferito al Font-Size, **number** è un numero puro (senza unità di misura e senza %), che produce una altezza pari a N * Font-Size.
Text-Decoration	< none | [underline | overline | line-through] >
Word-Spacing	<normal | <length> Necessaria unità di misura Definisce una spaziatura addizionale fra le parole. Sono ammessi valori negativi,.
Letter-Spacing	<normal | <length> Necessaria unità di misura Definisce una spaziatura addizionale fra i singoli caratteri. Sono ammessi valori negativi.
Text-Indent	**Indentazione della prima linea di testo.** Applicabile soltanto ai Block Elements <length> | <percentage> Default 0. <percentage> è riferito al valore di Width
Text-Transform	<none | capitalize | uppercase | lowercase> • **uppercase** (Converte in maiuscolo l'intero contenuto) • **lowercase** (Converte in minuscolo l'intero contenuto) • **capitalize (Converte in maiuscolo il primo carattere di ogni parola)**

width e height

Dimension di un block tag. Possono essere applicati a qualsiasi tag block, compresi **checkbox** e **radio buttons**.

I BORDI di un contenitore

Border-Style	[none | dotted | dashed | solid | double | groove | ridge | inset | outset] Stile dei Bordi. Si possono specificare fino a quattro valori, nel qual caso il primo valore è riferito al Top Border, il secondo al Right-Border, Bottom Border e Left Border E' possibile utilizzare le proprietà singole **Border-Top –Style, etc** su ciascun bordo
Border-Width (solo dopo aver impostato Border-Style)	Consente di specificare tutti quattro i bordi nel seguente ordine: top, right, bottom, left Specificando un solo parametro, il valore viene applicato a tutti quattro i bordi. Specificando 2 parametri, il 1° valore viene applicato a top/bottom, il 2° a left/right
Border- Color (solo dopo aver impostato Border-Style)	<color> Colore dei bordi. Per default viene assunto il valore della Proprietà COLOR. Può contenere 1 – 2 – 4 valori nel seguente ordine: top, right, bottom, left Specificando un solo colore, il valore viene applicato a tutti quattro i bordi. E' possibile utilizzare le proprietà singole **Border-Top –Color, etc** su ciascun bordo

Ognuna delle proprietà precedenti può essere applicata a ciascun singolo bordo:

Border-Top-Width	<thin | medium | thick | <length> > Spessore del bordo superiore. Solo valori positivi.
Border-Left-Width	Spessore del bordo sinistro
Border-Right-Width	Spessore del bordo destro
Border-Bottom-Width	Spessore del bordo inferiore

Border	Consente di impostare in un sol colpo width, style e color di tutti quattro i bordi. Es p { border : 1px solid black }

E' anche possibile esprimere le 3 proprietà iniziali (style width e color) per ogni singolo bordo:

Border-Top	<border-top-width> || <border-style> || <color> Consente di impostare in un sol colpo width, style e color del Top Border
Border-Left	Consente di impostare in un sol colpo width, style e color del Left Border
Border-Right	Consente di impostare in un sol colpo width, style e color del Right Border
Border-Bottom	Consente di impostare in un sol colpo width, style e color del Bottom Border

Lo SFONDO di un contenitore

La proprietà Background consente di impostare lo **sfondo di un Tag** (cioè il colore e/o un'eventuale immagine di sfondo). Applicabile al body e a tutti i tag di tipo BLOCK. **Tutti i tag hanno per default sfondo trasparente**

Background-Color	<color> | **transparent** > Colore di sfondo dell'elemento (nel caso l'img non venga caricata)
Background-Image	**< url(img/sfondo.gif) | none | initial** (valore originario)> Immagine di sfondo. **Se il nome del file contiene degli spazi occorrono gli apici** : `url('mio file.jpg');` **E' buona abitudine usare SEMPRE gli apici all'interno della URL**
Background-Repeat	<repeat | repeat-x | repeat-y | no-repeat> repeat = ripete l'immagine di sfondo sia orizzontale sia verticale riempiendo la pagina. Default repeat-x = ripete l'immagine di sfondo soltanto in orizzontale (tile orizzontale) repeat-y = ripete l'immagine di sfondo soltanto in verticale (tile verticale)
Background-Position	Specifica l'allineamento dell'immagine di sfondo rispetto al contenitore. **Il 1° valore indica il posizionamento orizzontale, il 2° indica il posizionamento verticale** Valori possibili: **LEFT/CENTER/RIGHT, TOP/CENTER/BOTTOM** Ad esempio `background-position:RIGHT BOTTOM`, Se si specifica un solo valore si intende orizzontale, ed il verticale è assunto = `center` Per **default** l'allineamento è LEFT TOP (0%, 0%)
Background-Attachment	<**scroll** | **fixed**> Con **scroll** l'immagine di sfondo scorre insieme al contenuto della pagina Con **fixed** l'immagine di sfondo rimane fissa sul video anche quando si fa scorrere il contenuto
Background	Consente di specificare tutte le proprietà di Background in un'unica dichiarazione

NB Il **path** dell'immagine **deve** essere sempre definito a partire dalla cartella nella quale si trova il file **.css**

NB: L'immagine viene **SEMPRE** visualizzata a partire dallo spigolo in alto a sinistra. Se il contenitore è molto più piccolo dell'immagine verrà visualizzata solo una piccola porzione di immagine (appunto la porzione in alto a sinistra). Volendo visualizzare una porzione diversa di immagine, **è possibile applicare a background-position dei valori negativi che sostanzialmente traslano l'immagine verso sinistra e verso l'alto.**
Ad esempio `background-position:-50px -50px` visualizza l'immagine a partire dalle coordinate (50,50)

Background-size	**auto** (default) se l'immagine è più grande del contenitore viene visualizzata solo una porzione in alto a sinistra **cover**: l'immagine viene ridotta in modo da ricoprire interamente il contenitore **contain**: l'immagine viene ridotta in modo da essere completamente visualizzata cover e contain differiscono solo se il contenitore ha un aspect ratio differente rispetto all'immaginee

MARGIN e PADDING

Margin rappresenta il margine **esterno** di un elemento. Consente di distanziare / avvicinare un elemento rispetto agli elementi vicini. Applicabile soltanto ai tag di tipo BLOCK, cioè ai contenitori.

Margin-Top **Margin-Left / Right** **Margin-Bottom**	<length> | <percentage> | < auto > Sono ammessi valori negativi. <**percentage**> è riferito alle dimensioni del genitore. Margini di elementi consecutivi si sovrappongono
Margin	Consente di specificare tutti quattro i margini nel seguente ordine: **top, right, bottom, left** Specificando **un solo parametro**, il valore viene applicato a tutti quattro i margini. Specificando **2 parametri**, il primo viene applicato a **top/bottom**, il secondo a **left/right** Specificando **tre parametri** il significato è il seguente**: top, right/left, bottom** **E' possibile utilizzare su left/right il valore AUTO per eseguire la centratura orizzontale**

Padding rappresenta invece lo "**spazio interno**" tra il bordo e il contenuto del tag.

Padding-Top **Padding-Left / Right** **Padding-Bottom**	Distanza tra il contenuto dell'elemento ed il bordo superiore. Accetta soltanto valori positivi. <percentage> è riferito alle dimensioni del genitore. **NON è CONSENTITO usare AUTO con PADDING**
Padding	Consente di specificare tutti quattro i padding esattamente come per **margin**.

Il padding può anche essere utilizzato per impostare l'indentazione del tag `blockquote`.

```
blockquote {  padding-left: 10px;  }
```

Le pseudoclassi

:hover stile generico (cioè applicabile a tutti i tag) dal **verbo inglese** hover che significa **svolazzare**.
Applicato in corrispondenza del mouse-over ed automaticamente rimosso in corrispondenza del moue-out.
Es `p:hover { color : yellow; }`

:focus stile applicato solitamente sui controlli indica la presenza del focus sul controllo.
Utile ad esempio per modificare lo stile di un campo di input nel momento in cui riceve il focus.
:valid applicabile ai soli textbox quando contengono un valore valido. Ad es un number compreso tra min e max
:invalid applicabile ai soli textbox quando contengono un valore NON valido

a:link stile applicabile solo al tag <a> rappresenta il colore del collegamento ipertestuale nello stato di riposo
a:visited stile applicabile solo al tag <a> rappresenta il colore del tag dopo essere stato 'visitato'
a:active stile applicabile solo al tag <a> rappresenta il colore del tag nel momento in cui viene cliccato

p:nth-of-type(1) Indica il primo tag p in una sequenza di fratelli

Notare che :`hover`, :`focus` etc sono applicabili ***anche*** ad un selettore di classe, mentre
`nth-of-type(1)` è applicabile soltanto ai selettori di elemento come spiegato in dettaglio a pagina 18.

Regole base di applicazione dei selettori

- I vari selettori di stile possono far riferimento a tag posizionati uno dentro l'altro (figli e nipoti).
  A tale scopo si utilizza l'operatore **SPAZIO**
  `#box p a` significa `tag a` inserito all'interno di un `tag p` inserito all'interno del `tag` `#box`
  `#box .red` significa elementi interni al tag #box che implementano la classe red
  `#box p.red` significa elementi di tipo **p** interni al tag #box che implementano la classe red
  `div.class1 div.class2` significa `tag div` con class2 definiti all'interno di `tag div` con class1
- Tra un selettore e l'altro si può utilizzare la **parentesi angolare** > per indicare un figlio diretto.
  `#box > p` significa `tag p` figlio diretto di `#box`
- Anche davanti al selettore assoluto e davanti ad un nome di classe si può facoltativamente inserire il nome del tag a cui si vuole fare firefimento: `div.myclass{}` : `div.#myId{}`
- Le regole di stile possono essere assegnate contemporaneamente a più selettori.
  A tal fine occorre separare i tag mediante l'operatore **VIRGOLA**
  `div, p, .riga` { } significa applicare lo stile a tutti i tag div, p e alla classe .riga
- Le varie regole possono essere arbitrariamente combinate insieme.
  `#liv1 label, #liv2 .class1, #liv2 p a, .riga a { }`
- **Uh qualsiasi elemento può implementare contemporaneamente più classi** separate mediante uno spazio. Es `<div class="box graphics">` In tal caso **NON conta** l'ordina con cui sono richiamate le classi, ma l'ORDINE con cui le classi sono scritte all'interno del file.CSS

Livelli di priorità

Gli stili vengono normalmente assegnati in cascata. A parità di importanza il successivo copre il precedente.
Esistono però 4 livelli di priorità che appunto prevalgono rispetto all'ordine di scrittura degli stili

Liv 4 **L'attributo !important introduce un livello 4 prioritario rispetto a TUTTI gli altri livelli**,
Liv 3 **I selettori di elemento (selettori assoluti) sono prioritari rispetto agli altri due selettori**,
Liv 2 **Le psudoclassi hanno lo stesso livello di priorità delle classi. Es: p:nth-of-type()**
Liv 2 **I selettori di classe sono prioritari rispetto ai selettore di tag**
Liv 1 **I selettori di tag hanno il livello di priorità più basso**

Specificità

In realtà il livello di priorità di un qualunque selettore (anche annidato) viene determinato sulla base della **specificità**

Ogni selettore ha un "peso" dato da tre livelli di specificità, indicati come una terna: `(a, b, c)` in cui i tre numeri indicano rispettivamente:

1. Numero di ID
2. Numero di classi/pseudoclassi
3. Numero di elementi

Si considerino i due seguenti esempi:

```
#liv1 .myclass
#liv1 div p a
```

`#liv1 .myclass`
- **#liv1** 1 ID selector
- **.myclass** 1 class selector

  Specificità: (1, 1, 0)

`#liv1 div p a`
- **#liv1** 1 ID selector
- **div**, **p**, **a** 3 element selectors

  Specificità: (1, 0, 3)

La priorità si valuta partendo dal primo numero, in caso parità si valuta il secondo, in caso di parità il terzo. Entrambi hanno **1 ID**, quindi si passa al livello successivo:

- (`#liv1 .myclass`) ha **1** classe
- (`#liv1 div p a`) ne ha **0** quindi ha priorità inferiore.

La proprietà DISPLAY : BLOCK

I tag **`block`** (**`div, p, h, li, table`**, etc) sono quelli che vanno a capo. Si tratti in genere di contenitori preposti a contenere testo o altri oggetti

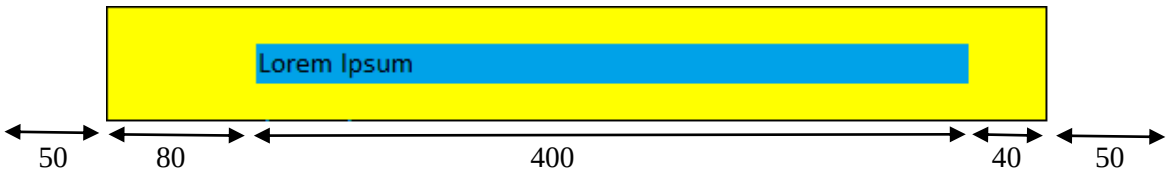
- riconoscono le proprietà **`width height`**.
  - il valore di default di **width** è **l'intera larghezza del genitore**
  - il valore di default di **height** è **fit-content**, cioè l'altezza necessaria a contenere il testo o gli elementi interni.
  - **`width`** accetta come valore anche **`fit-content (`**che imposta una larghezza pari al contenuto)
- I tag block vengono posizionati per default **uno sotto l'altro.** Anche quando **width** viene ridotto, continuano comunque ad 'impegnare' l'intera riga e non accettano altri oggetti a fianco (salvo utilizzo di `float`)
- Riconoscono **`Text-Align`**, ma non **`Vertical-Align`**

Occupazione complessiva di un BLOCK TAG

I valori di **width** e **height** **NON rappresentano la reale dimensione del box, ma l'area utile interna**.
L'occupazione complessiva è ottenuta sommando l'eventuale dimensione del **padding** e del **bordo**.

Cioè il **padding contribuisce ad aumentare le dimensioni dell'oggetto**. Esempio:

```
p{   width:400px;
     height:30px;
     padding: 20px 40px 20px 80px;  // indicato in giallo
     border: 2px solid black;
     margin: 50px;   }
```



La **Width** complessiva risulta **400** + **40** (right padding) + **80** (left padding) + **4** (bordi) = **524 px**.
La **Height** complessiva risulta Height + PaddingTop + PaddingBottom + Bordi = **30** + **20** + **20** +**4** = **74px**.

Note sulla proprietà Margin

Margin definisce il margine esterno all'oggetto al di la del bordo e non interviene nel computo delle dimensioni.

- **I margini orizzontali** (left e right) **di elementi consecutivi si sommano sempre.**
- **I margini verticali** (top e bottom) **di elementi verticalmente consecutivi :**
  - Nel caso comune di elementi statici di tipo block **si sovrappongono**
  - Nel caso di elementi inline-block, float, position:absolute, display:flex i margini si sommano
- **Il margin TOP del primo elemento** (block, inline, inline-block) **interno ad un contenitore collassa, cioè 'fuoriesce' dal contenitore e va a sovrapporsi a quello del contenitore**. Idem per il margin-bottom dell'ultimo elemento. Se però il contenitore ha un **border**, i margini dell'elemento interno rimangono confinati all'interno del contenitore. Inoltre il 'collasso' non avviene se il contenitore è definito con overflow:auto

Centratura verticale del testo all'interno di un Block Tag

Se il contenitore contiene una sola riga, il modo più semplice per eseguire la centratura verticale è quella di utilizzare la proprietà **`line-height`**, impostando il valore di **`line-height`** allo stesso valore di **height**.
Vale anche per gli **input**[type=text]. La proprietà Line Height indica l'interlinea, cioè l'occupazione verticale della singola riga. Il default è Line-Height = Font-Size, cioè per default il tag ha una altezza pari al Font-Size.

Nel caso invece di righe multiple, non si può ovviamente agire su **Line Height.**
In questo caso la soluzione migliore è quella di **omettere** height (**`height: auto`**) e agire sul **`padding`**.
Se non si indica esplicitamente il valore di **`height`**, il tag DIV avrà una altezza pari all'altezza del testo, più il **`padding`** superiore e inferiore che renderanno il testo perfettamente centrato.

La proprietà DISPLAY : INLINE

I tag **`inline`** (**`a, span, label, b, i, u, img`**) **vengono visualizzati in linea con il testo circostante.**

- possono essere inseriti all'interno di una riga come il normale testo
- non riconoscono **`Width`** e **`Height`** (che assumo il valore **auto**, che equivale a **fit-content**)
- riconoscono normalmente le proprietà **background-color** e **`Padding`**.
- riconoscono i **margini** laterali ma non riconoscono il valore "auto"
- non riconoscono **`Margin-Top`** e **`Margin-Bottom`**.
- non riconoscono **`Text-Align`** (che non è significativa). ma riconoscono **`Vertical-Align`** che rappresenta l'allineamento verticale rispetto ad una ipotetica linea di testo, con l'oggetto che:
  - nel caso di align = bottom, si espande al di sopra della linea del testo,
  - nel caso di align = top, si espande al di sotto della linea del testo.

La proprietà DISPLAY : INLINE-BLOCK

Un tag **inline-block** è un tag **INLINE** che però riconosce alcune proprietà dei tag BLOCK. In particolare riconosce :

- **`width`** e **`height`**
- tutti i **`margin`** (quindi anche Margin-Top e Margin-Bottom) ma non il valore margin:**auto**

I tag **`img, input, button, textArea, select`** sono di default **`inline-bloc`**k**.**
Rappresenta un'ottima tecnica per visualizzare **una sequenza di elementi affiancati**.

Allinemento verticale fra tag inline e inline-block

- **`vertical-align`** assume default differenti da tag a tag e, a parità di tag, assume valori differenti a seconda che sia inline o inline-block. Per avere allineamento verticale fra più tag **inline-block**, la soluzione migliore è quella di assegnare lo stesso valore di **`vertical-align`** (**`top`** oppure **`middle`**) a tutti gli elementi.
- Se non fosse sufficiente, è possibile assegnare ai singoli tag un piccolo **`vertical-align`** numerico in px. Valori positivi spostano l'elemento verso l'alto, valori negativi lo spostano verso il basso.
- oppure ancora si può assegnare `position:relative`, e poi agire sulla proprietà `top`

La proprietà DISPLAY : GRID

Il valore **display:grid** consente una visualizzazione simile a quella di una tabella HTML:

category:	cooking
title:	Everyday Italian
lang:	en
author:	Giada De Laurentis
year:	2005
price:	30.00

```
#wrapper {
    display: grid;
    width: fit-content;        // larghezza del wrapper. Va bene anche un valore fisso: width=350px
    grid-template-columns: 160px auto;                    // larghezza di ogni colonna
    grid-template-rows: 25px 25px 25px 25px 25px 25px;    // altezza di ogni riga
    grid-column-gap:8px;
    grid-row-gap:5px;
}
```

Il valore **grid-template-columns** può essere impostato al valore **fit-content** su una o più colonne
Il valore **grid-template-row** deve essere settato per ogni singola riga.
Se omesso le righe assumono l'altezza necessaria per visualizzare il contenuto.
Se presente, tutti gli elementi interni assumono quell'altezza di riga, compresi **checkbox** e **radiobutton**

Note E' possibile definire anche un rowspan ed un colspan

```
#wrapper div:nth-of-type(1) { grid-column: span 2 }    si espande su 2 colonne.
#wrapper div:nth-of-type(2) { grid-row: span 2}        si espande su 2 righe
```

Per gestire gli allineamenti, si possono utilizzare sugli elementi interni anche la seguente proprietà specifiche per il display:grid (alternative a text-align e vertical-align che comunque continuano ad essere riconosciute)

```
#wrapper div{
    justify-self:end;          /* allineamento orizzontale */
    align-self:center;  }      /* allineamento verticale   */
```

Entrambe queste proprietà possono assumere i valori `start/end/center.`

La proprietà Overflow

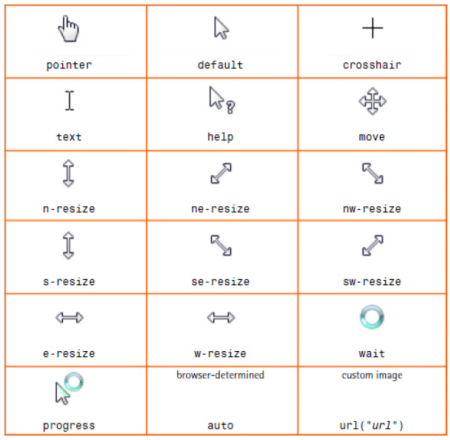
Ha senso quando il testo interno **eccede** le dimensioni di un contenitore che ha **dimensioni fisse**.

```
<div style="width:160px; height:80px; ">
```

I possibili valori che può assumere sono:

- **visible** (defult) La porzione eccedente le dimensioni del box viene mostrata eccedendo le dimensioni
- **hidden** La porzione eccedente le dimensioni del box non viene visualizzata
- **auto** Se si impostano Width e Height, viene visualizzata la barra di scorriemento necessaria (verticale o orizzontale o entrambe) ***solo*** se il testo eccede le dimensioni del contenitore.
  oveflow:auto è anche utilizzato, sempre su un contenitore, per "sentire" gli elementi FLOAT interni, nel qual caso occorre NON impostare height: il conteniotre si estende automaticaemnte
  oveflow:auto impostato su un contenitore, fa sì che l'elemento vada AUTOMATICAMENTE a colmare un eventuale spazio libero lasciato alla sua destra da un elemento float:left con una sua width.
- **scroll** Indipendentemente dal contenuto, sul contenitore vengono applicate sia la barra di scorrimento verticale che quella orizzontale. La barra di scorrimento orizzontale ha senso se all'interno del contenitore ci sono degli elementi (es immagini) la cui larghezza eccede le dimensioni del box, oppure degli elementi con position:absolute in posizioni che eccedono la larghezza del box.

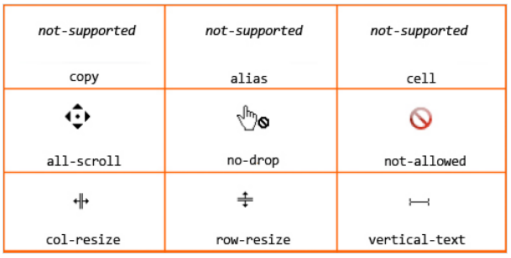
La proprietà cursor



Esempio:

```
div:hover { cursor: pointer; }
```

Nei CSS3 sono stati introdotti i seguenti cursori:



LIST Properties (Elenchi puntati e numerati)

List-Style-Type	disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | **none** Applicabile a **UL** e **OL**. Indica il tipo di puntino/numerazione da utilizzare `Esempio: OL { list-style-type: upper-alpha }`
List-Style-Image	<url> | none **Indica l'immagine da utilizzare come marker al posto dei puntini**. Prioritaria rispetto a List-Style-Type
List-Style-Position	inside | outside Inside fa sì che la seconda riga continui sotto il puntino
List-Style	Consente di impostare in un sol colpo le tre proprietà precedenti.

Ci sono alcune combinazioni per le quali **List-Style** viene automaticamente settato a **none**. Ad esempio :
1) quando gli LI sono di tipo inline-block
2) quando UL ha padding=0 e si trova ancorato sul lato sinistro del genitore.

CSS Reset

Spesso i tag utilizzano per le varie proprietà CSS valori di default differenti, il che spesso diventa fonte di problemi di configurazione. Non solo ma questi valori possono cambiare da browser a browser.

Per risolvere questi problemi spesso si utilizzano i cosiddetti ***file di reset*** tali da impostare gli stessi valori di default per tutti i tag (e per tutti i browser). Questi file sono detti ***CSS Reset*** e vanno inseriti **prima** di tutte le altre regole. Tra più utilizzati *Eric Meyer CSS Reset* e *Yahoo CSS Reset*. Tra i valori più importanti da resettare margin e padding

```
* {  margin: 0;   padding: 0;  ..... }
```

dove ***** rappresenta il cosiddetto **selettore universale** che indica "tutti i tag".

Altezza di un wrapper

Il valore di default dell'altezza di un qualunque tag Block è sempre **height:fit-content**. Ora se il contenitore principale (#wrapper) ha una altezza >= rispetto all'altezza dello schermo non ci saranno problemi.
Cioè la pagina html ed il suo sfondo si 'allungano' per tutta l'altezza dell'intero contenuto.

Se invece il contenitore principale (#wrapper) ha una altezza **inferiore** rispetto all'altezza dello schermo, la pagina html ed il suo sfondo si 'allungano' soltanto per l'altezza del contenuto, lasciando un colore di sfondo inferiore diverso piuttosto fastidioso. Per risolvere questo problema si possono impostare le seguenti property:

```
html, body, #wrapper {height:100%, margin 0}
```

dove **margin:0** evita che il **margine superiore** ecceda il contenitore creando una 'striscia 'superiore con il colore di sfondo del body (che potrebbe essere diverso rispetto al colore di sfondo di #wrapper). *Oppure* :

```
#wrapper {height:100vh}   // altezza del viewport. Estende anche body e html
```

Proprietà relative alle tabelle

border Può essere impostato **indipendentemente** sia sulla TABLE esterna (più spesso), sia sui TR (più sottile e più chiaro), ma anche sui singoli TD. Tutti questi tag hanno per default ***border:none***

border-spacing Impostato sul tag TABLE, definisce una possibile spaziatura tra TD adiacenti (all'interno della quale si vede il colore di sfondo della tabella). Default 2px

border-collapse: collapse; Indica che i bordi delle celle adiacenti devono "collassare" in un unico bordo
Con border-spacing:0 i due bordi si affiancherebbero, e dunque raddoppia lo spessore.

padding Non è definito sui tag TR ma solo sui tag TD

La proprietà FLOAT

La proprietà float consente di **ancorare un elemento su uno dei lati (destro o sinistro) del contenitore** in modo simile agli attributi HTML ALIGN=left **e** ALIGN=right del tag **IMG.**

- Gli elementi FOAT (come i position:absolute) non vengono conteggiati nel normale rendering della pagina, per cui **gli elementi BLOCK successivi** possono sovrapporsi all'elemento float
- La proprietà FLOAT è applicabile sia ai tag BLOCK sia ai tag INLINE che in ogni caso diventano **INLINE-BLOCK**, per cui **si estendono solo per la larghezza necessaria per visualizzare il loro contenuto,** a meno che non si specifichino le proprietà WIDTH e HEIGHT
- Negli elementi **FLOAT** (e position aboslute) i margini verticali non si sovrappongono ma si sommano
- Gli elementi **FLOAT** non **sentono margin:0 auto** per cui un tag float non può essere centrato nella pagina.

Comportamento dei tag successivi

- Elementi `float` successivi vengono affiancati
- Se gli elementi successivi sono di tipo **BLOCK ,** per evitare sovrapposizioni occorre assegnare ai Block Element successivi un apposita `width` ed un eventuale `margin-left`. Un elemento dichiarato `overflow:auto` viene affiancato all'elemento float

  L'elemento **block** successivo viene visualizzato sotto se implementa la proprietà `CLEAR:BOTH` la quale **interrompe la fluttuazione posizionando l'elemento sulla prima riga libera DOPO gli elementi fluttuanti**.
  Il valore CLEAR:LEFT interrompe soltanto il float:left e non il float:right e viceversa
  La proprietà CLEAR non ha effetto se assegnata su un tag inline o inline-block. Nel caso si può interporre un tag block vuoto

- Se gli elementi successivi sono di tipo **INLINE o INLINE-BLOCK** questi 'sentono' gli elementi float antecedenti, per cui verranno automaticamente posizionati al loro fianco senza sovrapporsi
  Ad esempio si può definire un elemento FLOAT:left seguito da un INLINE-BLOCK seguito da un FLOAT:right

  Allo stesso modo se l'elemento FLOAT è inserito all'interno di una porzione di testo, il testo semplicemente scorre intorno all'elemento FLOAT sul lato opposto rispetto all'elemento.

Comportamento del contenitore

Un contenitore di tipo Block non "sente" gli elementi Float, che dunque possono estendersi in verticale oltre la fine del contenitore, a meno che il contenitore non sia lui stesso `float` oppure `overflow:auto`. In entrambi i casi l'**altezza** del contenitore si adatta in modo da contenere tutti gli elementi float interni.

Per quanto concerne la **larghezza** del contenitore, nel caso di overflow:auto occupa tutta la riga, mentre nel caso float occupa soltanto lo spazio minimo necessario a contenere gli elementi float interni.
In entrambi i casi è bene specificare manualmente la larghezza desiderata.

La proprietà Position

Può assumere i seguenti valori:

- **`static`** [default]. L'oggetto segue il normale rendering della pagina, cioè il normale posizionamento HTML
- **`relative`** Impostando **`position:relative`** su un elemento, questo continua a rimanere '**statico**' però, tramite le proprietà **top, left, right** e **bottom** diventa possibile impostare un OFFSET rispetto alla normale posizione di rendering. Ad esempio top:10px sposta l'elemento di 10px in basso rispetto alla posizione che avrebbe con static. Viceversa bottom:10px consente di spostare l'elemento in alto di 10px. Sotto questo aspetto **`top:10px`** è equivalente a **`bottom:-10px.`**
  Se impostato su un contenitore, viene ereditato da **tutti** gli elementi interni.
- **`absolute`** : Impostando **`position:absolute`** su un elemento, questo non seguirà più il normale rendering della pagina ma, tramite le proprietà **top, left, right** e **bottom** diventa possibile eseguire un posizionamento assoluto dell'elemento rispetto all'intera pagina. Può anche capitare che, nella posizione stabilita, l'elemento vada a sovrapporsi ad altri elementi già presenti in quella posizione.
  I valori assunti per default sono top:0 e left:0 (angolo in alto a sinistra)
- Impostando **`position:absolute`** su un elemento interno ad un altro elemento con valore di **Position** diverso da static, (tipicamente **relative,** ma eventualmente anche **absolute** o **fixed),** il posizionamento degli elementi interni diventa ASSOLUTO rispetto al **contenitore** e non più rispetto alla pagina.
  In questi casi in genere si imposta sul contenitore il valore **`position:relative`** senza impostare **top** e **left** in modo che il contenitore segua il normale rendering della pagina consentendo però il posizionamento assoluto degli elementi interni.
- **`fixed`** Simile ad absolute, ma l'elemento risulta insensibile rispetto allo scroll della pagina.
  Inoltre non può essere riferito al contenitore ma è **SEMPR**E riferito alla pagina.

- Fra le quattro proprietà **top, left, right** e **bottom**, se ne impostano in genere soltanto due (top/left oppure bottom/right) e le altre due vengono desunte automaticamente sulla base di width e height.
- Nel caso di position:fixed è possibile impostare contemporaneamente top:0; left:0; right:0; bottom:0; per far sì che l'elemento **copra l'intera finestra** (viewport). Non bisogna però impostare **width** e **height**
- Gli elementi con `position absolute/fixed` non vengono conteggiati nel normale rendering della pagina, per cui più elementi possono sovrapporsi tra loro oppure possono sovrapporsi ad altri elementi statici presenti in quella posizione**.** La sovrapposizione dovrà essere gestita tramite z-index

- Negli elementi **`float`** e **`position!=static`** i margini verticali non si sovrappongono ma si sommano
- Impostando **`position!=static`** su un qualsiasi elemento, la proprietà **display** assume automaticamente il valore **inline-block**, per cui gli elementi possono affiancarsi

La proprietà z-index

Accetta un valore numerico compreso tra -2 Miliardi e +2 Miliardi. **Il default è zero**
Introduce, oltre a x e y, un terzo asse relativo alla profondità. Elementi con z-index maggiore vengono visualizzati davanti (cioè più vicini all'utente) rispetto ad elementi con z-index minore.

z-index è definito SOLTANTO per oggetti che hanno `position:absolute/relative/fixed`.
Tutti gli elementi statici hanno per default z-index=0 non modificabile.
Per visualizzare un elemento absolute al di sotto degli elementi statici occorre assegnargli il valore ***z-index = -1***

Nota:
Le proprietà **`position:abosolute`** e **`float`** sono in genere alternativa (o si imposta una oppure l'altra).
Nel caso si impostassero entrambe, la proprietà **`position:abosolute`** prevale sulla proprietà **`float`**.

Altre Proprietà CSS 2

max-width: larghezza massima dell'elemento
max-height: altezza massima dell'elemento
min-width: larghezza minima dell'elemento
min-height: altezza minima dell'elemento

`max-width` e **`min-width`** rappresentano un primo esempio di responsive design

- **max-width** serve per limitare la larghezza di un oggetto su schermi di grandi dimensioni.
- **min-width** serve per evitare un eccessivo 'rimpicciolimento' su schermi di piccole dimensioni

Entrambe queste impostazioni equivalgono sostanzialmente ad impostare ***`width:auto`***, cioè l'elemento si estende per l'intera larghezza del contenitore.

- Nel caso di **max-width** nel momento in cui la larghezza dell'elemento supera la soglia indicata, **width** assumerà il valore indicato, impedendo che l'elemento continui ad allargarsi oltre tale soglia
- Nel caso di **min-width** nel momento in cui la larghezza dell'elemento scende sotto la soglia indicata, **width** assumerà il valore indicato, impedendo che l'elemento continui a restringersi al di sotto di tale soglia

`max-height` risulta molto comodo nel caso in cui si abbia una sequenza di **tag img** disposti orizzontalmente , i quali adattano le loro dimensioni in base alle dimensioni delle immagini contenute. Invece di impostare una height fissa per tutte le img, si può utilizzare **`max-height`** in modo che le immagini più grandi vengano riscalate mantenendo le proporzioni, mentre le immagini più piccole verranno visualizzate così come sono senza doverle ingrandire con il rischio di perdere di qualità (ad esempio Esercizio Ricette Angular)

---

display Impostando il valore **non**e l'elemento viene completamente 'rimosso' dalla pagina e gli elementi successivi scorreranno verso l'alto occupando il posto dell'elemento con **`display:none`**.
Gli altri valori maggiormente utilizzati sono **`block, inline, inline-block, grid, flex`**

visibility meno forte del precedente. Può assumere i valori **visibility::hidden** **visibility: visibile** visibility:collapse. A differenza di **display:none**, nasconde l'elemento senza rimuovere la sua occupazione spaziale. Rimane un spazio bianco all'interno della pagina. E' come se rendesse l'elemento totalmente invisibile.
Sia **`display:none`** che **`visibilità:hidden`** **disabilitano gli eventi** !

opacity Ancora meno forte dei due precedenti. Numero con la virgola tra **0=Transparent** e **1=solido** (default). Utile per rendere semitrasparente un oggetto rispetto al contenitore sottostante.

- **A differenza dei due precedenti continua a sentire gli eventi** (**compreso :hover**).
- **Rende semi-trasparente sia il testo dell'oggetto, sia il suo sfondo.**
- **Applica inderogabilmente la trasparenza anche a TUTTI gli oggetti figli.** Se il box ha opacity:50% e l'elemento interno opacity:100%, l'elemento interno avrà opacità risultante 75%. Difficile da gestire.
- A differenza di RGBa **agisce anche alle immagini di sfondo.**
- Avendo un valore numerico, consente l'utilizzo della proprietà CSS3 **`transition`**.

rgba( ) Il meno forte di tutti. Può essere applicato **separatamente** al **testo** dell'oggetto oppure al suo **sfondo**
Esattamente come opacity è un numero decimale compreso tra **0=trasparente e 1=solido**

```
background-color: rgba(0, 0, 0, 0.50);  // oppure
color: rgba(0, 0, 0, 50%);
```

- **A differenza di `opacity`, il valore di RGBa NON viene ereditato dagli elementi interni.** MEGLIO, perché spesso gli elementi interni NON devono essere trasparenti.
- Riguardo alla funzione **rgb()** sono anche disponibili le seguenti sintassi:

```
color: rgb(0 0 0);          // senza virgole
color: rgb(0 0 0 / 0.5); // con trasparenza, senza ricorrere a rgba
```

user-select:none Applicata ad un **contenitore**, inibisce la selezione del testo negli elementi interni al contenitore.

white-space Può assumere i valori **normal** (default, il testo interno va a capo in corrispondenza del margine destro)
- **nowrap** viene eliminato l'a capo automatico e le righe lunghe debordano oltre il bordo destro del contenitore
- **pre** gli spazi multipli rimangono inalterati senza essere compattati, come il tag PRE dell'HTML. Le righe lunghe debordano
- **pre-wrap** come ***`pre`*** però con il wrap on (le righe lunghe non debordano ma vanno a capo).

Considerazioni sulle Unità di Misura

I campi Dimensionali (width, height, etc) possono essere espressi mediante le seguenti Unità di Misura:

Assolute

- **px** (pixel) (**1px = 3/4 pt**)
- **pt** (points; utilizzato solo per il font-size. **1pt=1/72 pollice**)
- **in** (inches; 1in=2.54cm)
- **cm, mm, pc** (picas; 1pc=12pt)

(Attenzione che tra il valore numerico e l'unità di misura **non devono essere lasciati degli spazi)**

Relative

- **%** rappresenta una percentuale **rispetto alla corrispondente proprietà del contenitore**
- **em** simile alla precedente, però **riferita al font-size dell'elemento corrente**

Riepilogo sulle Unità di Misura del font:

punti web	em	px	pt	
1	0,625 em	10 px	7,5 pt	xx-small
2	0,82 em	13 px	10 pt	x-small
3	**1 em**	**16 px**	**12 pt (default)**	**small**
4	1.13 em	18 px	13,5 pt (sono ammessi i decimali)	medium
5	1.5 em	24 px	18 pt	large
6	2 em	32 px	24 pt	x-large
7	3 em	48 px	36 pt	xx-large

`%  em  rem  vh`

% in genere è riferito al valore della stessa property del genitore
% applicato su **width,** indica le dimensioni dell'oggetto rispetto alla **width** del **contenitore.**
% applicato su **height** indica le dimensioni dell'oggetto rispetto alla **height** del **contenitore.**
% applicato a **font-size** indica un fattore di moltiplicazione del font-size **rispetto al font-size del contenitore**,
% applicato su **padding** e **margin,** indica le dimensioni rispetto alla **width** del **contenitore.**

em è sempre riferito al font-size dell'elemento stesso
em è una unita di misura che deriva dalla tipografia, dove **1em** rappresenta la larghezza della M maiuscola
em applicato a **font-size** equivale a % (rappresenta cioè fattore di moltiplicazione del font-size **rispetto al font-size del genitore).** **font-size:0.8em** equivale a **font-size:80%**
em applicato a **width height padding** e **margin,** indica le dimensioni rispetto al **font-size** dell'elemento stesso.
em applicato su **padding** e **margin,** indica le dimensioni rispetto alla **font-size** dell'elemento **stesso.**
Impostare `padding:1em` significa assegnare al paddding lo stesso valore del font-size.
Se l'elemento utilizza un font-size maggiore, il padding aumenterà di conseguenza. Idem se il font-size si riduce.

Nota: La property font-size viene di solito ereditata dal genitore (a differenza di padding, margin, e background-color che assumono il valore di default del tag in cui si trovano). **font-size:150%** equivale a **font-size:1.5em** e significa incrementare il font-size di un 50 % rispetto al font-size del genitore
I tag input hanno un font-size di default pari a **10pt**, indipendente dal font-size del genitore.

rem (root em) è identico a em ma è riferito al font-size del tag base <html>
rem applicato a **font-size** indica un fattore di moltiplicazione del font-size rispetto al font-size del tag <html>.
rem applicato a **width height padding** e **margin,** indica le dimensioni rispetto al font-size del tag <html>.

vh (Viewport Height): gestisce il dimensionamento di un elemento in relazione all'altezza della finestra del browser. L'unità vh è pari all'1% dell'altezza della viewport. Per rendere un elemento alto quanto l'intera finestra del browser si può impostare la sua height sul valore **100vh.** A differenza di height:100%, che deve essere applicato a tutti i genitori del tag corrente, 100vh estende l'elemento corrente e di conseguenza tutti i suoi genitori per l'intera altezza del viewport, cioè per l'intera altezza della finestra del browser.

Approfondimenti su selettori e pseudoselettori

Adjacent Selectors

- Il selettore > indica Figlio diretto. Il seguente indica tutti i tag p direttamente scritti all'interno dei tag div

  `div > p`

- Il selettore ~ indica ***tutti*** i fratelli successivo ad un certo tag.

  `div ~ p`

  Individua tutti i tag <p> che seguono genericamente un tag all'interno di un medesimo livello

- Il selettore + indica il ***primo*** fratello successivo.

  `div + p`

  Individua tutti i tag <p> che si trovano immediatamente dopo un tag (*directly following*).

Non esistono purtroppo selettori che consentano di accedere al fratello/i **antecedenti**.

Annidamento dei selettori

Data una struttura HTML del tipo

```
<div class="article">
      <p> Lorem Ipsum </p>
</div>
```

Gli stili dei tag **<p>** interni al tag **.article** possono essere scritti anche in modo annidato come segue:

```
.article{
      background-color:#F0F;
      p {          /* tutti i tag interni */
            background-color:#FFA;
      }
      > p {        /* solo i figli diretti */
            background-color:#AFF;
      }
}
```

Selettori di attributo [ ]

Il selettore di attributo **[ ]** consente di individuare tutti gli elementi della pagina che implementano un certo attributo html. E' utilizzabile **per qualsiasi attributo html**, compresi quelli definiti dal programmatore, **però SOLO per quelli statici definiti nel file HTML,** e non per quelli definiti dinamicamente da javascript.
Attenzione a **NON lasciare spazi** tra il nome del selettore e le parentesi quadre.

```
<input type="text" name="txtNome" classe="3B" required >
  input[type='text']
  input[type='text'][name='txtNome']  // possono essere applicati in sequenza
  input[name=txtNome]
  input[classe='3B']                  // attributo creato dal programmatore
```

Notare che i valori **`text, radio, txtNome, 3B`** possono essere scritti indifferentemente con o senza virgolette. Le virgolette diventano **obbligatorie** in caso di spazi o caratteri speciali (es punteggiatura).

Selettori di attributo per attributi booleani

Per gli attributi booleani invece delle [ ] si utilizza il simbolo **:** Sono però in tutto solo 5 :
Attenzione anche in questo caso a **NON lasciare spazi** tra il nome del selettore e i due punti.

- **:checked** *(radiobuttons o checkbox attualmente selezionati)*
- **:disabled** *(tutti i controlli aventi l'attributo html disabled=true)*
- **:enabled** *(tutti i controlli aventi l'attributo html disabled=false)*
- **:required** *(tutti i controlli aventi l'attributo required)*
- **:read-only**

I vari pseudoselettori possono anche essere applicati in forma combinata:

```
input[type=radio][name=opt1]:checked
```

Esempio: applicare uno sfondo grigio su tutti i textbox disabilitati

```
input[type="text"]:disabled {
    background-color: #ddd;
}
```

Lo pseudoselettore `nth-of-type(i)`

Quando, all'interno di un contenitore, si ha una sequenza anche NON consecutiva di elementi dello stesso tipo (ad es DIV, P o LI), è possibile accedere ad ogni singolo elemento della collezione utilizzando lo pseudo selettore **nth-of-type()** seguito dall'indice dell'elemento a cui si intende accedere:

```
#wrapper > div:nth-of-type(3)   // Terzo tag DIV figlio diretto di wrapper
#wrapper > p:nth-of-type(3)     // Terzo tag  P  figlio diretto di wrapper
```

`:first-of-type` è equivalente a `nth-of-type(1)`
`:last-of-type` è equivalente a `nth-of-type(ultimo)`

even e odd

Gli pseudoselettori :**`nth-child() e nth-of-type()`** oltre all'indice i-esimo accettano come parametro anche i valori: **`even`** tutti gli elementi pari - **`odd`** tutti gli elementi dispari.
`even` e **`odd`** **sono utilizzabili soltanto all'interno di** **`nth-child(even)`** e **`nth-of-type(odd)`**

Come si calcola nth-of-type

La scritta **`#wrapper p:nth-of-type(3)`** indica tutti quei tag **<p>**, figli o nipoti di wrapper, che godono della proprietà di essere **il terzo tag <p>** rispetto al proprio genitore.
La scritta **`#wrapper :nth-of-type(3)`** indica tutti quei tag, figli o nipoti di wrapper, che godono della proprietà di essere **terzogeniti del loro tipo** rispetto al proprio genitore.

Si tenga presente però che `nth-of-type` agisce SEMPRE SOLO a livello di tag, e non sulla base di classi o pseudoSelettori aggiuntivi.

Cioè tutti i vari pseudo selettori applicati ad un tag, anche in forma multipla, NON vengono applicati in cascata, ma **qualunque pseudoselettore viene applicato sempre sul tag iniziale**, e **NON** su una ipotetica collezione individuata dallo pseudoselettore precedente:

La scritta **`input[type=radio]:nth-of-type(5)`**
va a prendere il **5° tag input della pagina** (solo se questo tag è di tipo radio), **indipendentemente dal type dei tag input precedenti**. **Se ci fossero quattro `input[type=text]` e dopo un tag `input[type=radio]`, verrebbe restituito questo 5° tag input (di tipo radio)**

Altri esempi

- La scritta **`#wrapper p.myclass:nth-of-type(3)`** non rappresenta il terzo tag <p> che implementa la classe myClass, ma indica tutti quei tag **<p>**, figli o nipoti di wrapper, che godono in generale della proprietà di essere **terzogeniti del loro tipo.** Se questo terzogenito implementa la classe myClass, allora sarà restituito, altrimenti se il terzogenito non implementa la classe myClass, verrà restituito un insieme vuoto. **Cioè il 3 è calcolato sui tag <p>, non sulle classi myClass**
- La scritta **`#wrapper .myclass:nth-of-type(3)`** indica ***tutti*** i tag interni a wrapper che godono della proprietà di essere **terzogeniti del loro tipo,** ma ***solo se*** implementano la classe myClass.

Lo pseudoselettore `:nth-child(i)`

E' simile a **`nth-of-type(i)`** ma indica l'i-esimo elemento indipendentemente dal tipo.

`#wrapper` **`:nth-child(3)`**
indica il 3 elemento all'interno di wrapper indipendentemente dal suo tipo

`#wrapper p`**`:nth-child(3)`**
indica il 3 elemento all'interno di wrapper ma solo se è di tipo P

Pseudoselettori di tipo functions

```
:not(selettore_Secondario)   // Es input[type=radio]:not(:checked)
#menu li:has(ul)             // Le voci di menu che contengono il tag indicato (anche annidato)
:contains(testo);            // deprecato
```

Gli pseudoselettori `::after e ::before`

`::after` consente di aggiungere un testo in coda al contenuto dell'elemento corrente.
`::before` è simile ma aggiunge il testo PRIMA del contenuto del tag corrente.

Quello che in CSS2 era *`:after`*, in CSS3 è diventato *`::after`* perché, da un punto di vista logico, a differenza degli altri pseudoselettori, non rappresenta un elemento del DOM a cui accedere, ma consente di aggiungere un nuovo contenuto ad un certo elemento. Per cui è stata creata una sintassi differente :: che però da punto di vista sintattico è equivalente alla vecchia sintassi CSS2

In pratica è come se aggiungesse un tag span fittizio in coda o in cima al contenuto del tag.
Un apposito attributo content specifica il testo da aggiungere:

```
p::after {
  content: " - Remember this";
  background-color: yellow;
  color: red;
  font-weight: bold;
}
```

Il testo **`- Remember this`** sarà aggiunto in coda al contenuto di ogni tag P ed avrà sfondo giallo e testo rosso:

I live in Ducksburg - Remember this

Accesso agli elementi che implementano più classi

Qualunque tag html può implementare più classi:

```
<p class="class1 class2"> </p>
```

Se si vogliono definire delle proprietà CSS per gli elementi che implementano sia classe1 che classe2, nel file CSS le due classi devono essere scritte "attaccate" senza spazi intermedi

```
.class1.class2 {
    background-color:green;
}
```

Utilizzo dell'ID in modo non univoco

L'ID in realtà può anche essere assegnato in modo non univoco purchè su tag differenti (approccio non proprio ortodosso). Ad esempio se un tag DIV ed un tag P hanno entrambi `ID="titolo",` da CSS si può accedere separatamente ai due tag nel modo seguente:

```
div#titolo { color="blue" }
p#titolo { color="green"  }
```

Note su tag H e font-size

I tag H **anziché ereditare normalmente il font-size del genitore in cui si trovano, "modificano" il font-size ereditato applicando un fattore moltiplicativo o di riduzione** come indicato nella seguente tabella:

- **H1** raddoppia entrambi i valori (font **24pt** e padding **32px**)
- **H2** applica un aumento del 50% (font **18pt** e padding **24px**)
- **H3** applica un aumento del 16,5% (metà di 33) (font 14pt e padding19px)
- **H4** applica al font lo stesso font-size e lo stesso padding del genitore
- **H5** applica una riduzione di 16,5% (font 10pt padding 13px)
- **H6** applica una riduzione di 33% (font 8pt padding 10px)

Nota: Se il tag H1 anziché trovarsi dentro un tag DIV si trova dentro uno dei nuovi tag HTML5 (**`section article, nav, aside`**), il fattore moltiplicativo di H1 diventa solo più 1,5.
Per fare in modo che H1 si comporti sempre allo stesso modo indipendentemente dal tag genitore, si può impostare all'interno del file reset.css la seguente property: **`h1 {font-size:2em}`**
cioè h1 deve *sempre* raddoppiare il font-size del genitore, indipendentemente dal contenitore in cui si trova.

Il significato del valore `inherit`

Assegnare ad una Property CSS di un certo elemento il valore inherit, significa che quella property in quell elemento deve ereditare il valore della stessa property nell'elemento genitore (contenitore esterno), che peraltro rappresenta il comportamento di default di molti tag (tutti quelli relativi al testo, compreso color).
Questo comportamento non vale invece per padding, margin, background-color, border che, indipendentemente dal genitore, utilizzano il valore di default per quel tag. L'impostazione del valore inherit ha senso per queste proprietà che non ereditano automaticamente oppure per le proprietà il cui valore è stato modificato tramite CSS.

```
body { color:black;  font-family:Georgia; }
h2  { color:violet;  font-family:Arial;  }
#sidebar h2 { color: inherit;  font-family: inherit;  }
```

`h2` all'interno di `sidebar` non sarà violet ma eredita il valore di sidebar il quale a sua volta sarà ereditato da body, quindi il suo colore sarà **nero**. Idem per font-family.

Le seguenti assegnazioni sono invece inutili in quanto h3 eredita automaticamente da body

```
h3 { color: inherit;  font-family: inherit; }
```

Esempi di Effetti

Scroll morbido ad una ancora

```
html{  scroll-behavior:smooth; }
```

Questa proprietà fa si che, quando si clicca su un collegamento ipertestuale per raggiungere un'ancora di una pagina, lo scroll verso l'ancora non venga eseguito istantaneamente ma venga eseguito in modo graduale tramite uno scorrimento.

Posizionamento di una Barra a fondo pagina insensibile allo scroll

Si utilizza un primo DIV container largo quanto la pagina, avente **position:fixed** (che lo rende insensibile allo scroll) e **bottom:0** che lo posiziona allineato al bordo inferiore del genitore (la pagina HTML).
Si utilizza quindi un innerObject contenente gli oggetti desiderati.

```
#footer_bar {
      position: fixed;
      bottom: 0;
      width: 100%;
}
```

```
#bar_innerObject {
    background: #FAA53A;
    width: 900px;
    height: 35px;
    margin: 0 auto;
    overflow: hidden;   }
```

Eliminazione del bordo blu intorno alle immagini-link

Una immagine utilizzata come collegamento ipertestuale viene di solito visualizzata con un bordo blu intorno. Questo perchè i browser impostano di default il bordo blu sui link. Per eliminare il bordo occorre impostare :

```
img { border: none;  }
```

Inserimento di un testo all'interno di una immagine

Per ottenere l'effetto di un testo sovrapposto ad una immagine si può sfruttare la proprietà **background** inserendo l'immagine come sfondo del blocco testuale;

```
<div class="imgStyle2"> Lorem Ipsum</div>
```

```
div.imgStyle2 {
      width: 400px;
      height: 300px;
      background-image: url('img.jpg');
      background-repeat: no-repeat;
      text-align: center;
      line-height: 450px; /*225px superiori rispetto al centro scritta*/
}
```

All'interno del block DIV verrà caricata l'immagine di sfondo che deve avere le stesse dimensioni del contenitore oppure verrà visualizzata soltanto la porzioni di immagine in alto a sinistra con dimensioni pari alle dimensioni del box . Impostando **line-height > width** la scritta verrà visualizzata nella parte inferiore dell'immagine.

Inserimento di un testo a comparsa all'interno di un'immagine

```
<div class="comparsa">  <p> Lorem Ipsum </p>  </div>
```

```
div.comparsa {
      width: 400px;
      height:300px;
      background-image: url('img.jpg');
      background-repeat: no-repeat;
      position:relative;
      padding: 0;
}
div.comparsa p {
          background: rgba(0,0,0,0.5);  // 0=trasparente, 1=solido
          color:#EEEEEE;   /* colore del testo */
          display:none;    /* nasconde la scritta  */
          position:absolute;
          left:0px;
          bottom:0px;
          margin: 0;
          height:40px;
          line-height:40px;
          padding-left:10px;   /* piccolo spaziatura del testo a sinistra */
          width:390px;
}
```

Invece di impostare **width:390px** si potrebbe impostando **width: 100%** in modo che il tag <p> occupi la stessa larghezza del genitore. Avendo però impostato un padding di 10, (ed essendo width espresso al netto del padding) questo padding andrà a sommarsi alle dimensioni dell'area del tag p, eccedendo le dimensioni del contenitore. Il problema può essere risolto impostando sul contenitore esterno **overflow: hidden.** In questo modo, se le dimensioni del tag p interno eccedono le dimensioni del contenitore (come effettivamente è), la parte eccedente non viene mostrata.

Per far comparire il testo in corrispondenza del mouse over è sufficiente impostare;

```
div.comparsa:hover p {
   display: block;  }
```

Personalizzazione della grafica di checkbox e radio buttons

Per poter modificare la grafica di un checkbox o di un radio button, occorre necessariamente rimuovere gli stili applicati di default che non possono essere sovrascritti nemmeno con il **!important**

```
input[type=checkbox] {
        -webkit-appearance: none;
        display: inline-block;
        width:16px;  height:16px;
        padding:0;   border-radius: 3px;
        background-color:#eee;
        border: 1px solid #ccc;
        box-shadow: 1px 1px 2px rgba(0,0,0,0.05);
        position: relative;  }
input[type=checkbox]:checked {
        background-color: #ffa;
        color: #999;  }
input[type=checkbox]:checked::after {
        content: '\2714';  /* codice unicode della spunta */
        font-size: 12px;
        position: absolute;  top: -2px;  left: 2px;  }
```